

A Programmable Logic Controller (PLC)

The brain of industrial automation



What is?,uses and
common programming languages

A Programmable Logic Controller (PLC) is a ruggedized industrial computer that monitors inputs from sensors and switches, makes decisions based on a custom program, and controls output devices like motors and valves to automate manufacturing processes and other industrial functions. PLCs are designed for high reliability and operation in harsh industrial environments.



- Key Components:

A typical PLC system is comprised of several core components working together:

1. Central Processing Unit (CPU):

The "brain" of the PLC, which executes the control program, processes data, and manages communication.

2. Power Supply:

Converts incoming AC voltage to the stable DC voltage required by the internal circuitry of the PLC and its modules.

3. Input/Output (I/O) Modules:

These are the interfaces connecting the PLC to external field devices.

Input modules receive signals from sensors and switches, while output modules send control signals to actuators and other machinery. They can handle either digital (on/off) or analog (variable range) signals.

4. Memory Unit:

Stores the operating system, the user-defined control program (often in non-volatile memory), and data from inputs and program execution.

5. Communication Interface:

Enables connectivity with other systems, such as Human-Machine Interfaces (HMI), Supervisory Control and Data Acquisition (SCADA) systems, or other PLCs, using various industrial protocols like Modbus or Ethernet/IP.

6. Programming Device:

A computer or handheld device with specialized software (e.g., Siemens' TIA Portal, Omron's Sysmac Studio) used to write, debug, and download the control program to the PLC.

- How PLCs Work:

PLCs operate in a continuous loop called a scan cycle that typically involves four stages:

1. Read Inputs:

The PLC checks the status of all connected input devices (e.g., is a switch on or off?) and stores this data in its memory.

2. Execute Program:

The CPU runs the user-defined logic program step-by-step, using the input statuses in memory to determine what the outputs should be.

3. Update Outputs:

The PLC changes the state of the output devices (e.g., turning on a motor or light) based on the results of the program execution.

4. Housekeeping:

The PLC performs internal diagnostics and communication tasks before starting the cycle again.

This entire cycle happens very quickly (in milliseconds), allowing for real-time, high-speed control of industrial processes.

- Types and Applications:

PLCs come in various types tailored to different needs:

1. Fixed/Compact PLCs:

All-in-one units with a fixed number of I/O points, ideal for small, cost-sensitive, or standalone applications like simple conveyor systems.

2. Modular PLCs:

Consist of a chassis/rack that accepts separate modules (CPU, power supply, I/O, communication) for high scalability and customization, suitable for large, complex operations like automotive assembly lines or chemical processing plants.

3. Safety PLCs:

Specialized controllers designed for safety-critical functions, with features like redundancy and self-diagnostics to meet safety standards (e.g., managing emergency stop buttons).

PLCs are used across virtually all industries, including manufacturing, energy and utilities, food and beverage, water treatment, and transportation, for tasks such as controlling robotic arms, managing packaging lines, and regulating temperature and pressure in industrial processes.

- PLC common programming languages:

The five common programming languages for Programmable Logic Controllers (PLCs) are defined by the international standard IEC 61131-3. A comprehensive PLC programmer should be familiar with all five to choose the best one for a specific application.

- The five languages are:

1. Ladder Diagram (LD)
2. Structured Text (ST)
3. Function Block Diagram (FBD)
4. Sequential Function Chart (SFC)
5. Instruction List (IL)
6. Ladder Diagram (LD)

- Description:

The most common and widely used PLC language, **LD is a graphical language that resembles electrical relay circuit diagrams**. Its format uses two vertical "rails" with horizontal "rungs" that contain logic instructions.

*** Best for:**

Simple control applications, boolean logic (on/off tasks), and tasks involving interlocks and basic sequences.

*** Advantages:**

Easy to learn, especially for electricians and technicians familiar with relay logic.

Simple to monitor and troubleshoot due to its visual nature.

*** Disadvantages:**

Becomes unwieldy and hard to manage for complex programs.

Less suitable for complex mathematical calculations or algorithms.

- Structured Text (ST):

*** Description:**

A high-level, text-based language similar to programming languages like C, C++, or Pascal. It is known for its flexibility and power.

*** Best for:**

Complex mathematical calculations, data manipulation, algorithms, and applications requiring advanced control logic.

*** Advantages:**

Intuitive for programmers with a software development background.

Can handle complex tasks and algorithms that are difficult to express in graphical languages.

*** Disadvantages:**

Can be harder for maintenance technicians without a programming background to understand and troubleshoot.

More challenging to debug than graphical languages.

- Function Block Diagram (FBD):

*** Description:**

A graphical language that uses interconnected "blocks" to represent functions, with lines defining the data flow. It's useful for creating reusable sections of code.

*** Best for:**

Continuous process control, PID loops, analog scaling, and motion control.

*** Advantages:**

Provides a visual overview of a program's structure. Excellent for modular programming and reusing code blocks.

*** Disadvantages:**

Can become disorganized and hard to read in large-scale implementations.

Less common in some industries compared to Ladder Logic.

- Sequential Function Chart (SFC):

*** Description:**

A graphical language that organizes programs into sequential steps, making it ideal for modeling processes that occur in a specific order. It is similar to a flowchart.

*** Best for:**

Complex, multi-state processes, batch production, or systems with linked processes that need to run sequentially or in parallel.

*** Advantages:**

Simplifies complex sequential processes by breaking them into manageable steps.

Easy to monitor and debug since the process state is clearly visible.

*** Disadvantages:**

Limited to sequential applications and not ideal for pure logic control.

Can be complex for beginners to learn.

- Instruction List (IL):

*** Description:**

A low-level, text-based language that resembles assembly code. It uses mnemonic codes for instructions like "Load" and "Store".

*** Best for:**

Highly specific, low-level control logic and tasks that require very compact, time-critical code.

*** Advantages:**

Provides maximum flexibility and granular control for advanced users.

*** Disadvantages:**

Deprecated for new projects under the IEC standard. Difficult to read and debug compared to modern languages.

Less commonly used in current PLC programming.

By: Haytham Zeidan

<https://archive.org/details/@wazefapress>

Resources:

<https://inductiveautomation.com/resources/article/what-is-a-PLC>

<https://claroty.com/glossary/programmable-logic-controller-plc>

<https://erieit.edu/introduction-to-programmable-logic-controllers/>

https://en.wikipedia.org/wiki/Programmable_logic_controller

<https://eshop.se.com/in/blog/post/complete-guide-to-programmable-logic-controllers.html>

<https://industrial.omron.eu/en/products/programmable-logic-controllers>

<https://www.sciencedirect.com/science/article/abs/pii/S1364032116000551>

<https://www.eaton.com/in/en-us/products/controls-drives-automation-sensors/programmable-logic-controllers/understanding-programmable-logic-controllers.html>

<https://tulip.co/blog/programmable-logic-controller-what-is-a-plc/>

<https://www.lselectricamerica.com/blog/programmable-logic-controllers/>

<https://www.cryotos.com/glossary/plc>

<https://www.uti.edu/blog/robotics-and-automation/what-is-a-programmable-logic-controller>

<https://www.realpars.com/blog/plc-programming-languages>

<https://petrotechinc.com/why-is-the-iec-61131-the-standard-programming-language-in-open-architecture-control-systems/>

<https://plcsimulator.online/docs/iec61131-3>

https://en.wikipedia.org/wiki/IEC_61131-3

<https://inductiveautomation.com/blog/plc-programming-languages-go-beyond-ladder-logic>

https://en.wikipedia.org/wiki/Ladder_logic

<https://industrialautomationco.com/blogs/news/common-types-of-plc-programming-languages>

<https://www.plcdepartment.com/blogs/plcblog/plc-programming-languages-a-comparison-of-ladder-logic-function-block-diagrams-and-structured-text>

https://cdn.automationdirect.com/static/helpfiles/ls_plc/Content/A_IntroductionTopics/LP001-2.htm

<https://www.plcacademy.com/function-block-diagram-programming/>

<https://controlbyte.tech/blog/ladder-logic-what-is-it-and-how-does-it-compare-to-other-plc-programming-languages/>

<https://www.scribd.com/document/719360850/PLC-language>

<https://controlsoft.ca/plc-programming/plc-programming-languages/>

<https://www.realpars.com/blog/ladder-logic-vs-other-languages>